Lab 1: Installing Docker

Step 1: Uninstall old versions

Older versions of Docker were called `docker` or `docker-engine`. If these are installed, uninstall them, along with associated dependencies.

sudo rpm -qa | grep docker

sudo yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-engine

Step 2: Set up repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

Step 3: Install docker engine

Install the *latest version* of Docker Engine and containerd, If prompted to accept the GPG key, verify that the fingerprint matches 060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35, and if so, accept it. Docker is installed but not started. The `docker` group is created, but no users are added to the group.

sudo yum install docker-ce docker-ce-cli containerd.io

Verify the software installation using the rpm command

rpm -qa | grep docker

Step 4: Start docker and verify

sudo systemctl start docker

docker info

docker images

Congrats !! you have successfully completed the lab.

Lab 2: Docker fundamentals

Step 1: Docker status

Before running docker commands, check the status of docker daemon whether it's running.

systemctl status docker

```
[root@dockerhost ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Sun 2021-02-07 07:26:38 UTC; 19min ago
     Docs: https://docs.docker.com
 Main PID: 8446 (dockerd)
   Memory: 74.3M
   CGroup: /system.slice/docker.service
           └─8446 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Step 2: Verify docker

Use docker info to verify both client and server installed and operational

docker info

```
[root@dockerhost ~]# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with BuildKit (Docker Inc., v0.5.1-docker)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
```

Step 3: Check docker image

Verfiy whether we have any docker images available by default

docker images

Pull the centos and nginx image from the default repository and verify the same whether they are downloaded or not

docker pull centos

docker pull nginx

docker images

```
[root@dockerhost ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
nginx        latest    f6d0b4767a6c   3 weeks ago    133MB
centos       latest    300e315adb2f   2 months ago   209MB
```

Step 4: Inspect the resources
Use the docker inspect command to see details related to the images you just downloaded.

```
[root@dockerhost ~]# docker inspect centos
[
    {
        "Id": "sha256:300e315adb2f96afe5f0b2780b87f28ae95231fe3bdd1e16b9ba606307728f55",
        "RepoTags": [
            "centos:latest"
        ],
        "RepoDigests": [
            "centos@sha256:5528e8b1b1719d34604c87e11dcd1c0a20bedf46e83b5632cdeac91b8c04efc1"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2020-12-08T00:22:53.076477777Z",
```

Step 5: Run your first container
Using the docker run command run your first container.

docker run hello-world
The command will download the hello-world image from the repository and then will run the container.

```
[root@dockerhost ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:31b9c7d48790f0d8c50ab433d9c3b7e17666d6993084c002c2ff1ca09b96391d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Step 6: Verify container
Use the docker ps command to verify the container state

docker ps

```
[root@dockerhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@dockerhost ~]# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED              STATUS                          PORTS     NAMES
e6ba0639bf93   hello-world   "/hello"   About a minute ago   Exited (0) About a minute ago             quirky_banzai
[root@dockerhost ~]#
```

Step 7: Deploy detach mode container
Deploy the nginx container in detach mode and verify whether it is running or not.

docker run -d nginx

```
[root@dockerhost ~]# docker run -d nginx
8f58f4fb8aa4f5cb6ac7e59c2837e4088ea6ab622a52a41ea62829881a9d299e
[root@dockerhost ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
8f58f4fb8aa4   nginx     "/docker-entrypoint.…"   3 seconds ago   Up 2 seconds   80/tcp    upbeat_solomon
[root@dockerhost ~]#
```

docker ps
To verify the container running or not.

Step 8 : Login into container
Once the container is in running state, login into the container using the docker exec command.

docker exec -it containername *"/bin/bash"*

```
[root@dockerhost ~]# docker exec -it upbeat_solomon "/bin/bash"
root@8f58f4fb8aa4:/#
```

Logout from the container and delete the container

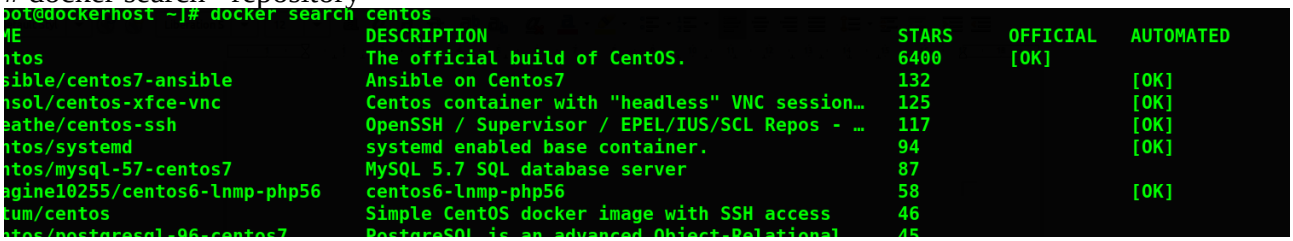
logout
docker stop containername
docker rm containername

Congrats !! You have successfully completed the lab

Lab 3: Docker image operations

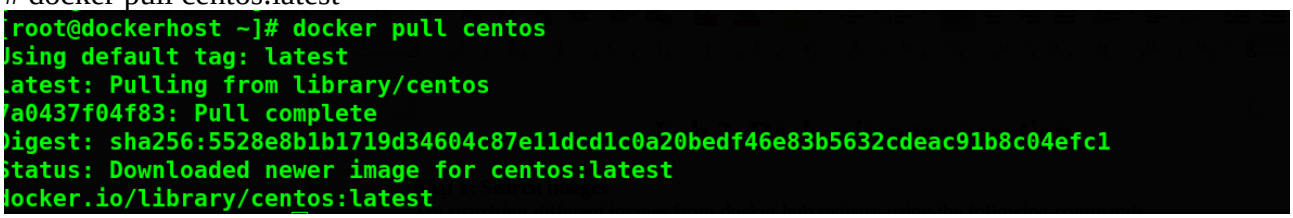
Step 1: Search images
Try searching different images from docker hub registry using the following commands

docker search <repository>



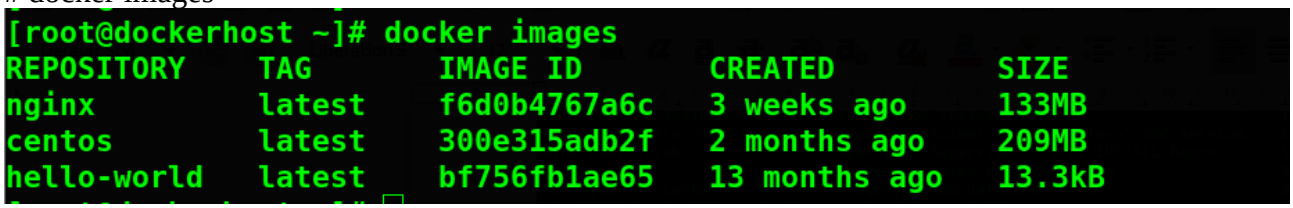
Step 2: Pulling image
Download the centos image from the repository

docker pull centos:latest



Step 3: Verify image
Check whether the docker image is downloaded or not

docker images



Step 4: Remove image
Download the hello-world image and after verify remove the same.

docker pull hello-world
docker images
docker rmi hello-world

Step 5: Commit image
Run the container with nginx image and login into the container, create directory inside the container and write data inside it. Logout from container and create a new image from container with docker commit command.

docker run -d nginx
docker exec -it <contname>

inside the container # mkdir /dirdata
inside the container # echo “hello there” > /*dirdata/newfile*
inside the container # logout

docker ps
docker commit <contname> mycustomimage:latest
docker images

Step 6: Deploy new container
Deploy the container with newly created container and verify the data in it

docker run -d mycustomimage
docker exec -it <contname>

inside the container # ls /dirdata

Step 7: Saving image
Save the image you created in step 5 into tar file named myimg.tar

docker images
docker save -o myimg.tar mycustomimg

Chapter 4: Docker file

Step 1: Make a directory

mkdir /test

cd /test

Step 2: Dockerfile

Now create a file `**Dockerfile**` (File name is hard coded do not change the file name)

touch /test/Dockerfile

Step 3: Sample Index.html

vim /test/index.html

insert the data in it like:
Welcome to customize image using dockerfile

Step 4: Edit Dockerfile

vi Dockerfile

FROM centos:latest

MAINTAINER NewstarCorporation

RUN yum -y install httpd

COPY index.html /var/www/html/

CMD [“/usr/sbin/httpd”, “-D”, “FOREGROUND”]

EXPOSE 80

Step 5: Build Image

Build the image using docker build. Test is the directory where Dockerfile is present and –t option is to tag or name the image.

#docker build /test/ -t webserver:v1

Step 6: Run the container

docker run -d webserver:v1

Chapter 5: Docker volume and Network

Step 1: Volume create
The docker volume create command will create a named volume. The name allows you to easily locate and assign Docker volumes to containers.

sudo docker volume create --name data-volume

Step 2: Container with volume
To launch a container which will use a volume that you have created with docker volume, add the following argument to the docker run command:

docker run -it --name my-volume-test -v data-volume:/data centos /bin/bash

You are logged into the container, check the mount by listing directories inside the container

inside the container # ls /

Step 3: Inspect the volume
To inspect a named volume, use the command:

docker inspect docker-volume

Step 4: Remove the volume
To remove a named volume, use the command:

sudo docker stop my-volume-test
sudo docker rm my-volume-test

Step 5: Create container with host directory volume
You can test this by first creating a directory to use as a Docker volume with the command:

sudo mkdir /hostvolume

Add a small test file to this directory with the command:

echo "Hello World" >> /hostvolume/host-hello.txt

Next, launch a container named my-directory-test and map /hostvolume on the host to /containervolume on the container with the command:

docker run -it --name my-directory-test -v /hostvolume:/containervolume centos /bin/bash

Once you are at the new container's command prompt, list the files in the shared volume with the command:

inside the container# ls /containervolume.

You will see the host-hello.txt file which we created on the host.

Step 6: List network
Each Docker installation automatically builds three default networks. You can list these networks with the command:

docker network ls

Step 7: Inspect Network
Inspecting a Docker network will return information about that network, including which containers are attached to the network, and their IP addresses. This is a valuable testing tool, as well as an easy way to find your container's IP address

docker network inspect <networkname>

Step 8: Create bridge network
To create your own network, the command is

docker network create -d bridge new_test_network

Step 9: Container with new network
If you do not specify a network when you start up a container, it will automatically be added to the default bridge network.

To start up a container and add it to a specific network, use the command:

docker run -it --net=new_test_network --name test_centos_container centos

Use CTRL-pCTRL-q to detach from the container.

You can use sudo docker network inspect [network name] to check the network and verify that the container is attached.